Introduction to Artificial Intelligence

丁尧相
浙江大学

Fall & Winter 2022
Week 5

Announcements

- We will release Problem Set 1.3 after this lecture.
- Problem Set 1 will due on next Monday.
- We will release the first lab project this week (MCTS).

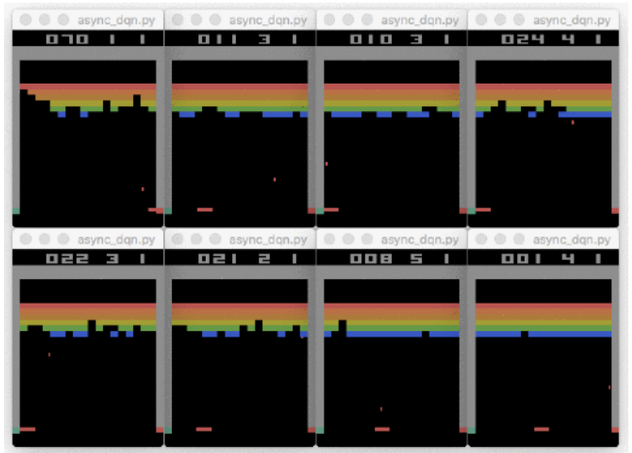
Decision Making: III

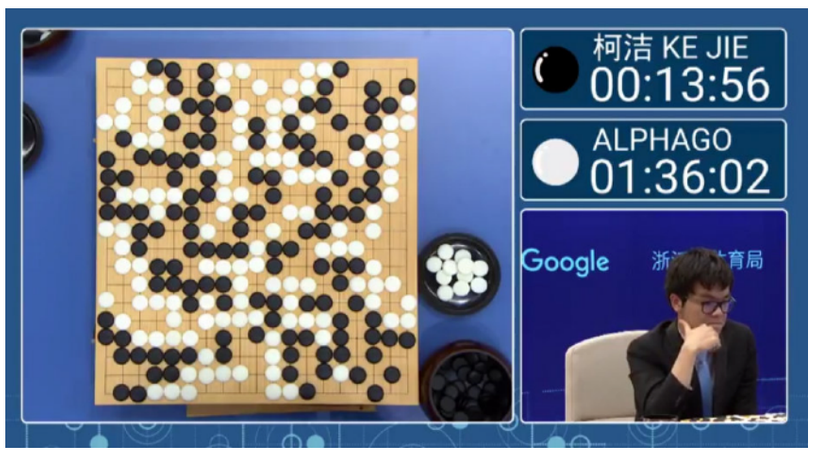
- Reinforcement learning (RL): background
- Markov decision process
- RL with known model
- RL with unknown model
- Take-Home Messages

Decision Making: III

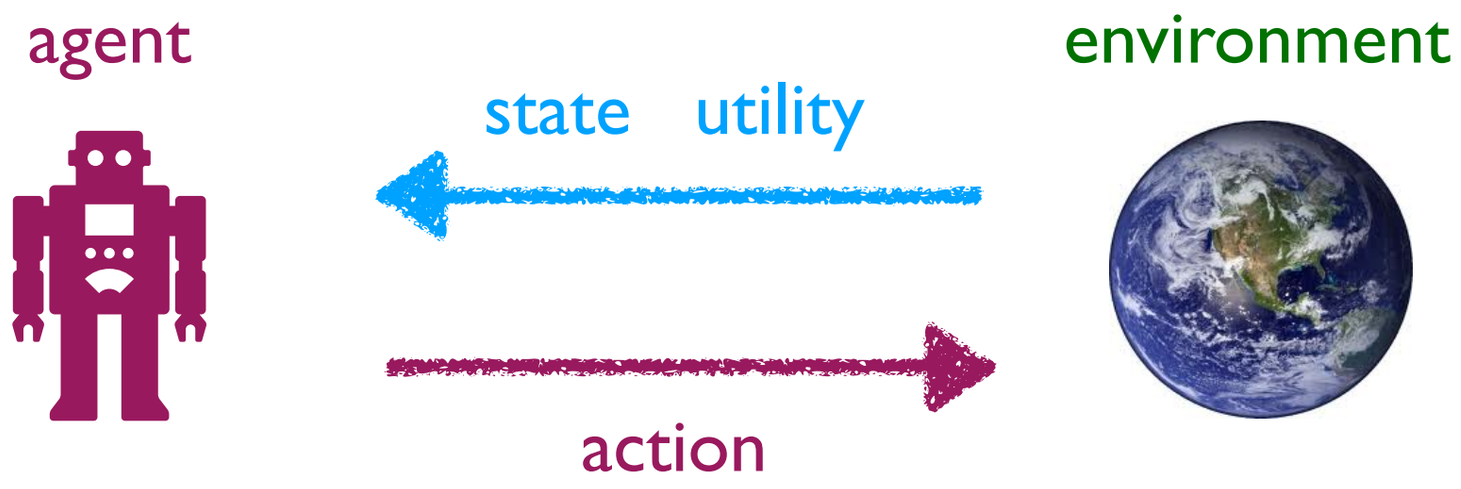
- Reinforcement learning (RL): background
- Markov decision process
- RL with known model
- RL with unknown model
- Take-Home Messages

Decision Making



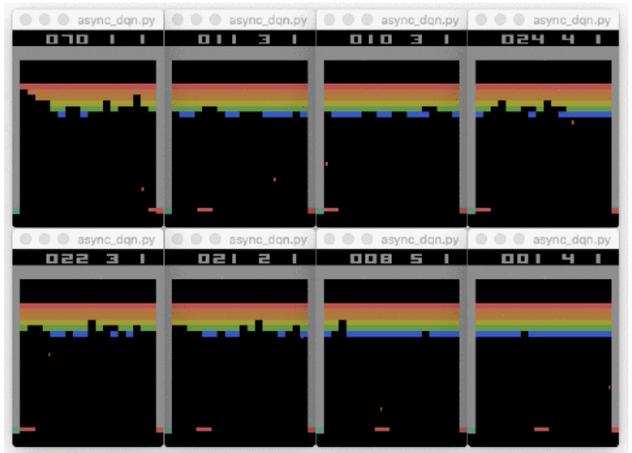


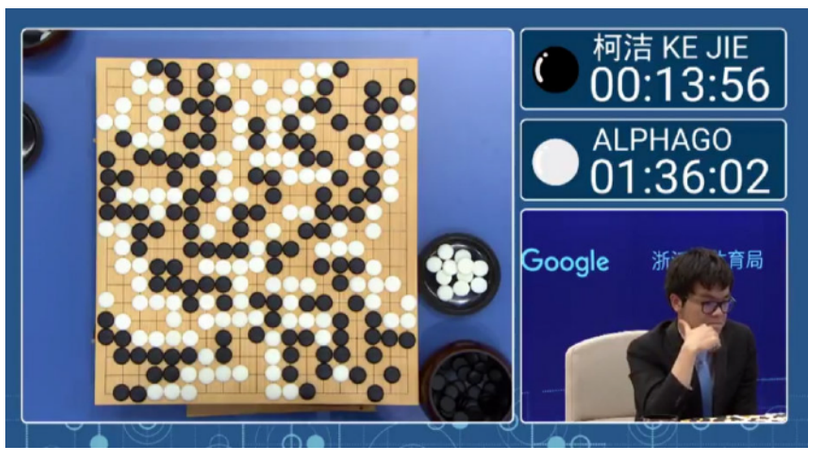
- Conduct action in any state of an environment.



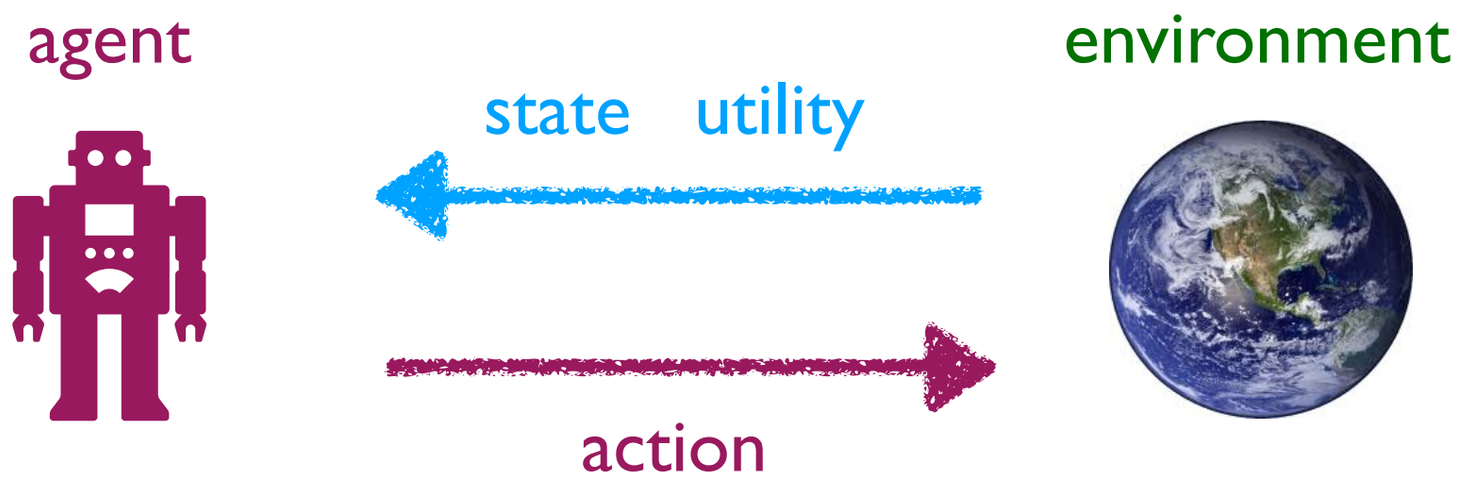
In most problems, the agent needs to do a sequence of actions w.r.t. a sequence of states.

Decision Making



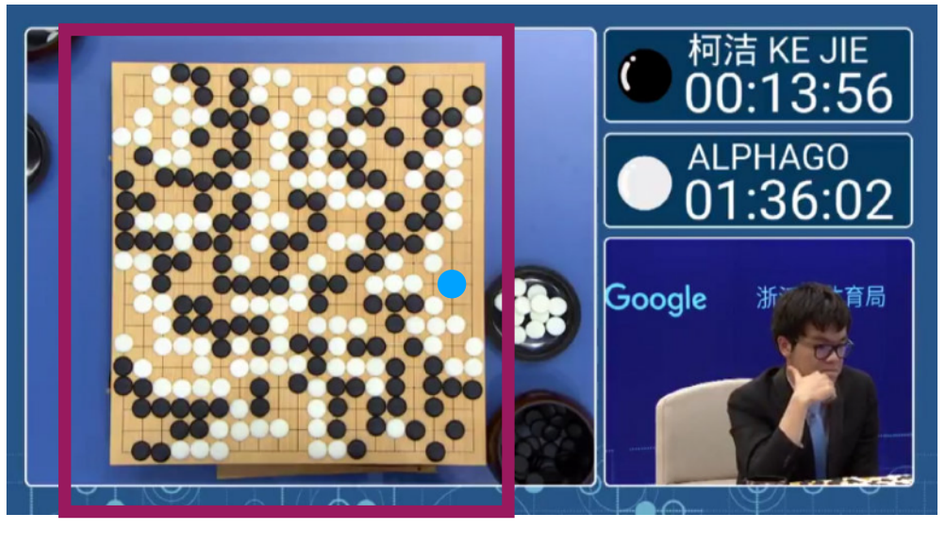


- Conduct action in any state of an environment.



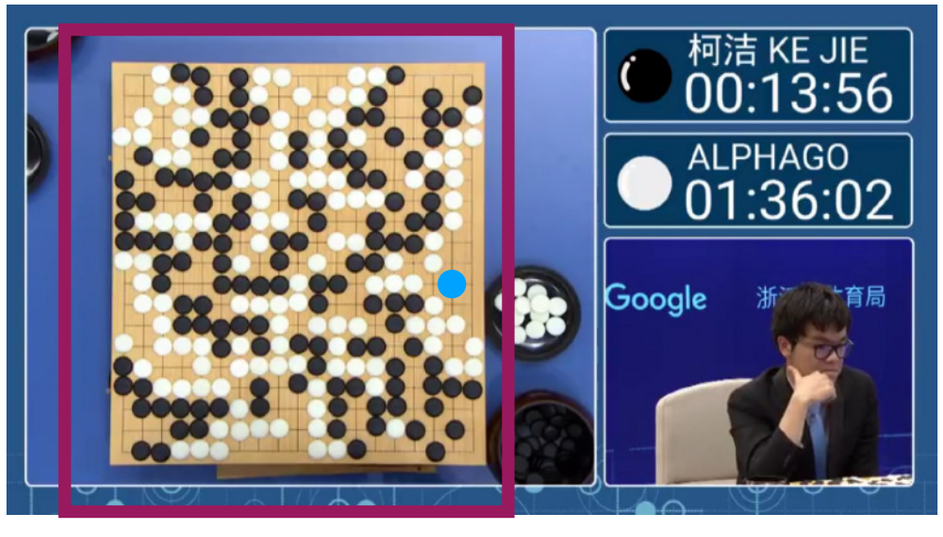
In most problems, the agent needs to do a sequence of actions w.r.t. a sequence of states.

The Decision Problem



- The agent faces with a series of "states".
- Need to choose the corresponding "actions".
- Each action has a utility/cost.
- Target: maximize the total reward in a decision sequence by always choosing the right action.

The Decision Problem

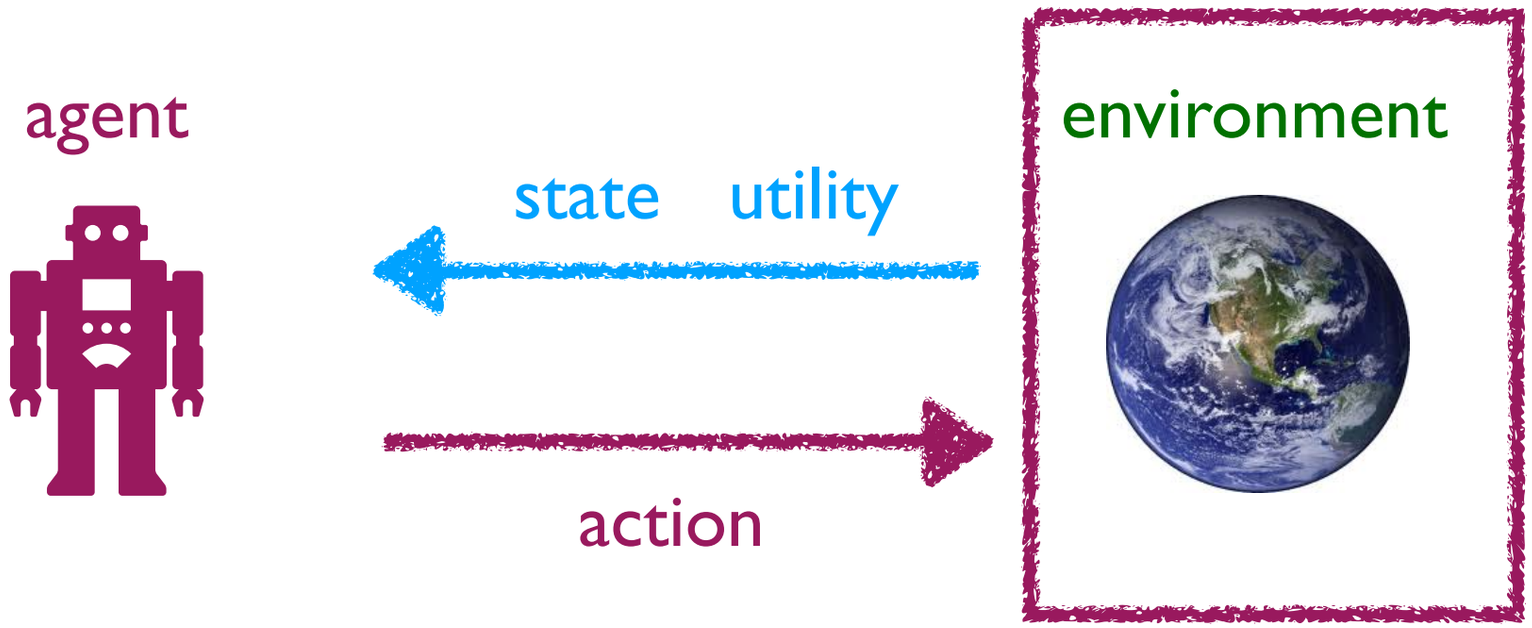


- The agent faces with a series of "states".
- Need to choose the corresponding "actions".
- Each action has a utility/cost.

In this lecture,
we given utility a name: reward.

- Target: maximize the total reward in a decision sequence by always choosing the right action.

Model of the Environment



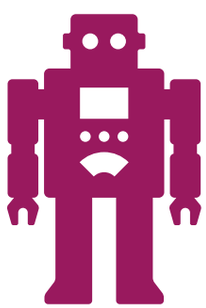
To make decisions in the environment, the agent usually needs a model of the environment to know how the things go on.
Where does this model come from?
Given by the problem (external) or built by the agent? (internal)

Internal vs. External Model

In the past two decision-making lectures, we assume that the agent can directly use the external model. We will consider building the internal model in the knowledge reasoning lectures!

internal model

agent



partial state

utility

action

environment (external model)

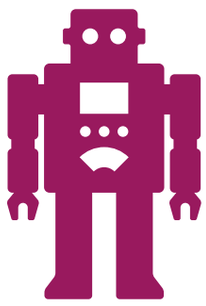
Internal vs. External Model

In the past two decision-making lectures, we assume that the agent can directly use the external model. We will consider building the internal model in the knowledge reasoning lectures!

But what if both the external and internal models can not be used?

internal model

agent



partial state

utility

action

environment (external model)

Reinforcement Learning

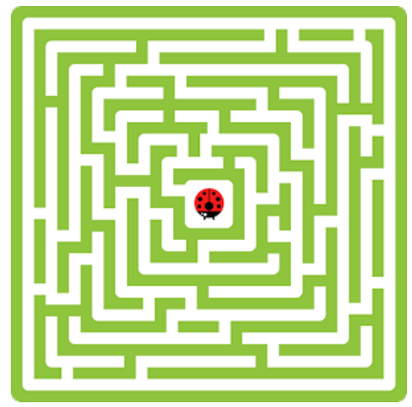
- Decision making is to find the optimal policy:
    - Decide best actions on all states.
    - No labeled <state, action> data, only receive reward.
        - The target is to maximize the long term total reward.
- Search-based decision making:
    - When the model is known, and the search cost is reasonable.
- Reinforcement learning:
    - Decision making in unknown model or search cost is too high.

Decision Making: III

- Reinforcement learning (RL): background
- Markov decision process
- RL with known model
- RL with unknown model
- Take-Home Messages

Markov Decision Process

- How to model a maze problem?



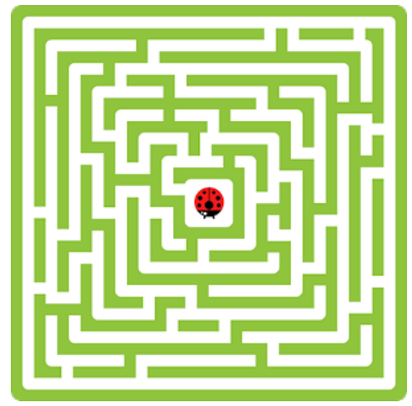
- State: the current position.
- Action: left, right, up, down.
- Transition: where is the next position when take an action?
- Reward: how good is it instantly when take an action?
- Discount factor: How much the current action influences future?

Markov decision process (MDP) is the decision making model in RL with specific assumptions.

Mathematical Formulation

- A Markov Decision Process (MDP) is a five-tuple

$$< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$$



- $\mathcal{S}$ — The space of possible states (cont. or discrete)
- $\mathcal{A}$ — The space of possible actions (cont. or discrete)
- $\mathcal{P} : p(s_{t+1}|s_t, a_t)$ — The transition function (distribution)
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ — The reward function
- γ — The discount factor of rewards

The transition and reward functions can be stochastic!

The Markov Property

"The future is independent of the past given the present"

- The Markov property:

$$p(s_{t+1}|s_t, a_t) = p(s_{t+1}|s_1, s_2, ...s_t, a_t)$$

- The next state is only decided by the current state and action.
- The current state is a sufficient statistic.
- Non-Markovian decision problem:

小张出生于中国，2016年来到美国留学。小张的母语是(？)

The Learning Agent

- The agent takes a series of actions, experiences a series of states, and receives a series of rewards:

$$\{s_1, a_1, r_1\}, \{s_2, a_2, r_2\}, \{s_3, a_3, r_3\}...$$

- policy: function $p(a|s)$ used to select actions on any states.

- The target is to find the optimal policy to maximize the discounted total reward along the timeline:

$$r_1 + \gamma r_2 + \gamma^2 r_3 + ... = \sum_{t=1}^{\infty} \gamma^{t-1} r_t$$

- The discount factor measures how much the long term effect of the current action is concerned.

Value Functions: State Value Function V

- The state value function of a given policy is the expected total reward start from a given state, then follow the policy:

$$V_\pi(s) = \mathbb{E}_{\pi, p(s|s,a)}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s\right]$$

- The optimal policy have the optimal value function:

$$V_{\pi^*}(s) = \max_\pi V_\pi(s)$$

Value Functions: Action-State Value Function Q

- The action-state value function of a given policy is the expected total reward start from a given state, execute a given action, then follow the policy:

$$Q_\pi(s,a) = \mathbb{E}_{\pi, p(s|s,a)}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a\right]$$

- The optimal deterministic policy chooses the optimal action:

$$\pi^*(s) = \arg\max_a Q_{\pi^*}(s,a)$$

If the optimal action-state value function is known, so is the optimal policy!

Bellman Equation

- For the state value function,

$$V_\pi(s) = \mathbb{E}_{\pi(s),p(s|s,a)}\Big[\sum_{t=0}^{\infty}\gamma^t r_t|s_0 = s\Big]$$

$$= \mathbb{E}_{\pi(s_0),p(s_1|s_0,a_0)}\Big[r_0 + \gamma\mathbb{E}_{\pi(s),p(s|s,a)}\big[\sum_{t=1}^{\infty}\gamma^{t-1} r_t|s_1\big]|s_0 = s\Big]$$

$$= \mathbb{E}_{\pi(s_0),p(s_1|s_0,a_0)}\Big[r_0 + \gamma V_\pi(s_1)|s_0 = s\Big]$$

Recursive Definition

- For discrete state and action, and deterministic policy,

$$V_\pi(s) = \sum_{s'} p\big(s'|s,\pi(s)\big)\Big[r\big(s,\pi(s),s'\big) + \gamma V_\pi(s')\Big]$$

Bellman Equation (Cont.)

- For the action-state value function,

$$Q_\pi(s,a) = \mathbb{E}_{\pi(s),p(s|s,a)}\Big[\sum_{t=0}^{\infty}\gamma^t r_t|s_0=s,a_0=a\Big]$$

$$= \mathbb{E}_{\pi(s_0),p(s_1|s_0,a_0)}\Big[r_0+\gamma\mathbb{E}_{\pi(s),p(s|s,a)}\big[\sum_{t=1}^{\infty}\gamma^{t-1}r_t|s_1,a_1\big]|s_0=s,a_0=a\Big]$$

$$= \mathbb{E}_{\pi(s_0),p(s_1|s_0,a_0)}\Big[r_0+\gamma Q_\pi(s_1,a_1)|s_0=s,a_0=a\Big]$$

Recursive Definition

- For discrete state and action, and deterministic policy,

$$Q_\pi(s,a) = \sum_{s'} p\big(s'|s,a\big)\Big[r(s,a,s')+\gamma Q_\pi\big(s',\pi(s')\big)\Big]$$

Bellman Equation (Cont.)

- For optimal deterministic policy π^*,

$$V_{\pi^*}(s) = \max_a \Big[\sum_{s'} p(s'|s,a)[r(s,a,s') + \gamma V_{\pi^*}(s')] \Big]$$

$$Q_{\pi^*}(s,a) = \sum_{s'} p(s'|s,a) \Big[r(s,a,s') + \gamma \max_{a'} Q_{\pi^*}(s',a') \Big]$$

- Then the optimal deterministic policy is

$$\pi^*(s) = \arg\max_a Q_{\pi^*}(s,a)$$

- Due the recursive structure, the optimal value functions can be solved by dynamical programming. This assumes that the full information of the MDP is known!

Decision Making: III

- Reinforcement learning (RL): background
- Markov decision process
- RL with known model
- RL with unknown model
- Take-Home Messages

Value Iteration

- Initialize value function V_0
- For i=1,2,3… until convergence
    - Update V_i for each state

$$V_i(s) = \max_a \Big[\sum_{s'} p(s'|s,a)[r(s,a,s') + \gamma V_{i-1}(s')] \Big]$$

- Theoretical convergence guarantee to V^* and π^*

Value Iteration

- Initialize value function V_0
- For i=1,2,3… until convergence
    - Update V_i for each state

Why iterative update?
Loop exists in the MDP!

$$V_i(s) = \max_a \Big[\sum_{s'} p(s'|s,a)[r(s,a,s') + \gamma V_{i-1}(s')] \Big]$$

- Theoretical convergence guarantee to V^* and π^*

Policy Iteration

- Initialize value function V_0 and policy π_0
- For i=1,2,3… until convergence
    - Policy evaluation step: update V_i for each state

$$V_i(s) = r + \gamma V_{i-1}^{\pi_{i-1}}(s')$$

    - Policy improvement step: update π_i for each s-a pair.

$$\pi(s) \leftarrow arg\max_a \sum_{s',r'} p(s', r|s, a)[r + \gamma V(s')]$$

- Theoretical convergence guarantee to V^* and π^*

Policy Iteration

- Initialize value function V_0 and policy π_0
- For i=1,2,3… until convergence
    - Policy evaluation step: update V_i for each state

$$V_i(s) = r + \gamma V_{i-1}^{\pi_{i-1}}(s')$$

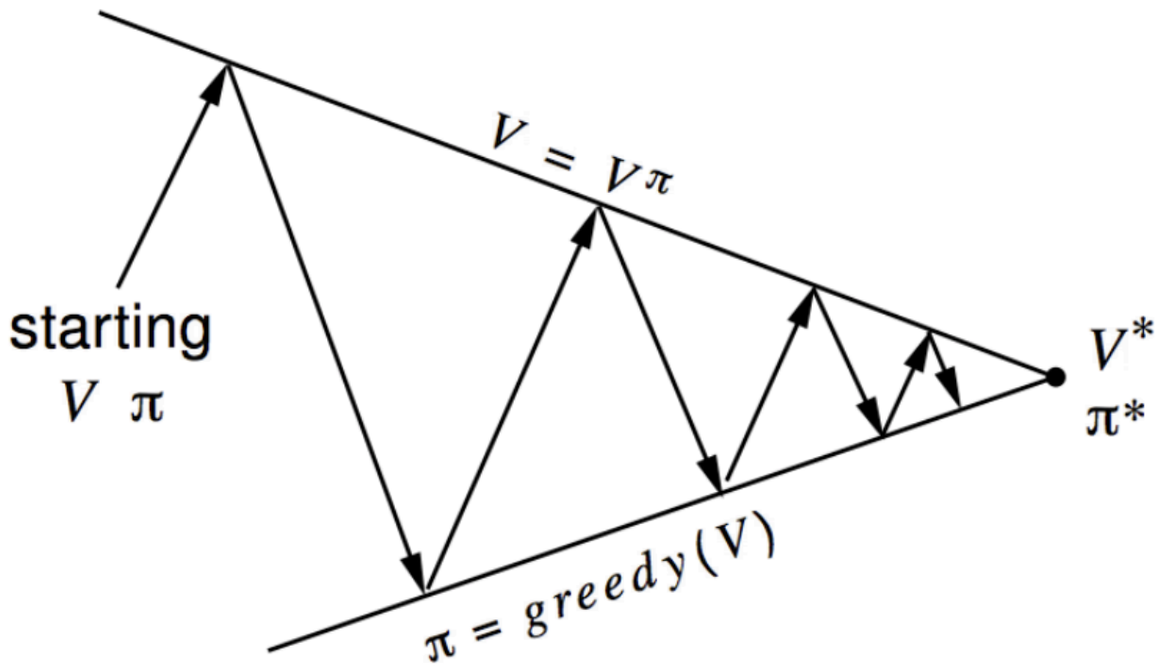
Calculated based on π_{i-1}
Actually an inner loop to do iterative update until convergence

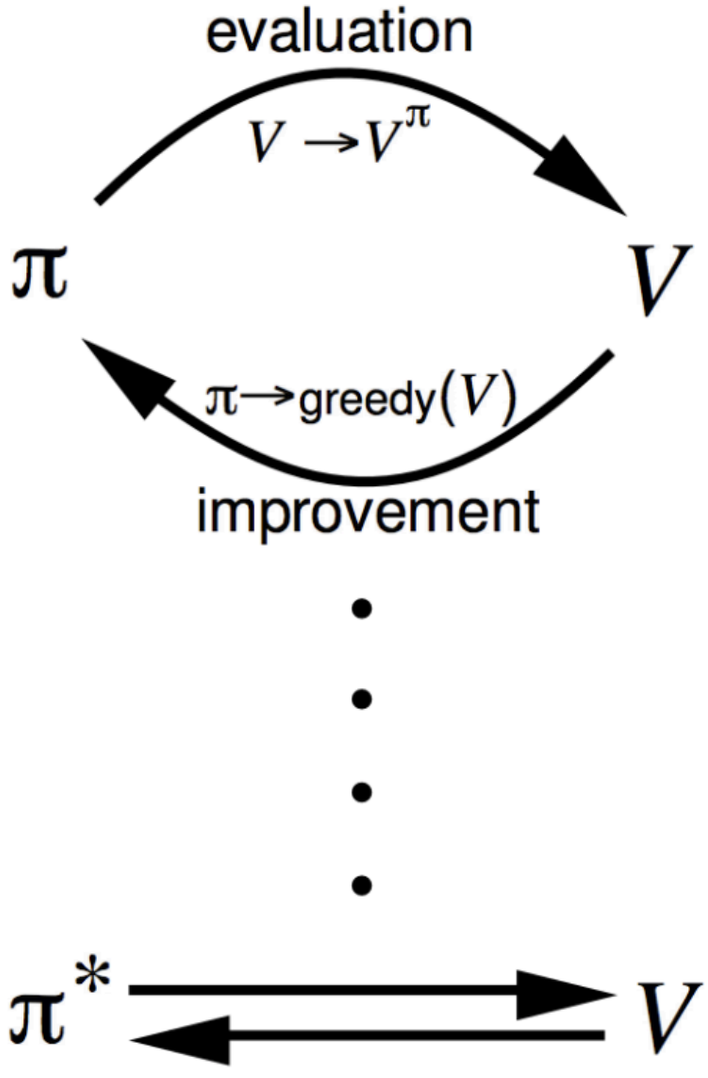
    - Policy improvement step: update π_i for each s-a pair.

$$\pi(s) \leftarrow arg\max_a \sum_{s',r'} p(s', r|s, a)[r + \gamma V(s')]$$

- Theoretical convergence guarantee to V^* and π^*

Policy Iteration (Cont.)





Policy evaluation Estimate v_π
Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$
Greedy policy improvement

Slide courtesy: David Silver

Short Recap

- The sequential decision problem can be modeled as an MDP.
- The Bellman equation describes the recursive structure of the value functions.
- When full information of the MDP is known, dynamical programming based planning, such as policy and value iteration, can solve for the optimal value function.
- When deal with large state-action space, the value function can be approximated, such as by linear functions:

$$Q(\phi(s,a)) = w^T \phi(s,a)$$

theoretical guarantee holds

Least square policy iteration (LSPI)

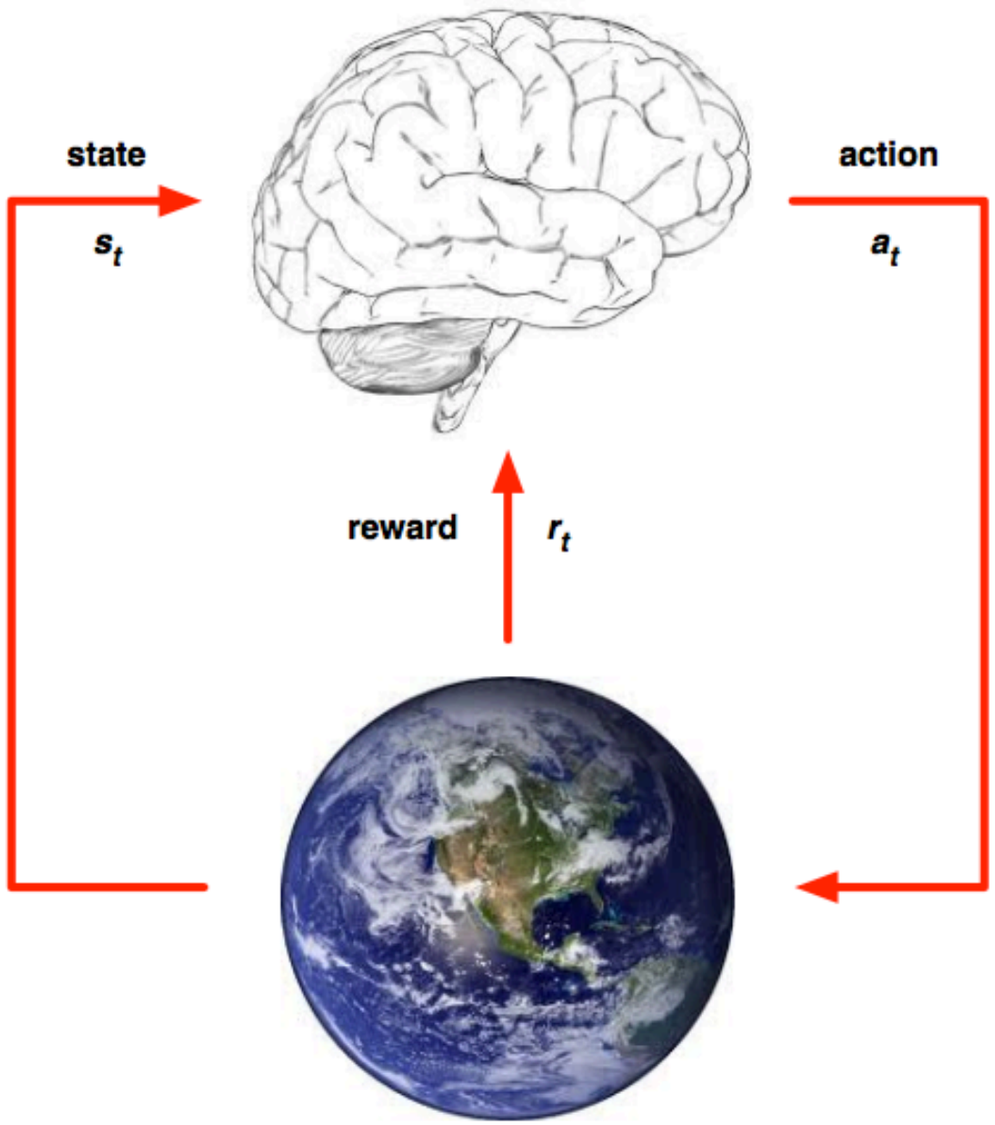
Decision Making: III

- Reinforcement learning (RL): background
- Markov decision process
- RL with known model
- RL with unknown model
- Take-Home Messages

Reinforcement Learning

- When full information of the MDP is known, the value function can be solved by planning.
- But how to solve when not fully known? $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$
- In RL, usually the state transition $\mathcal{P}$ and reward function $\mathcal{R}$ are not known.
- The agent has to learn by trial and error, facing with the exploration and exploitation problem.

Agent and Environment

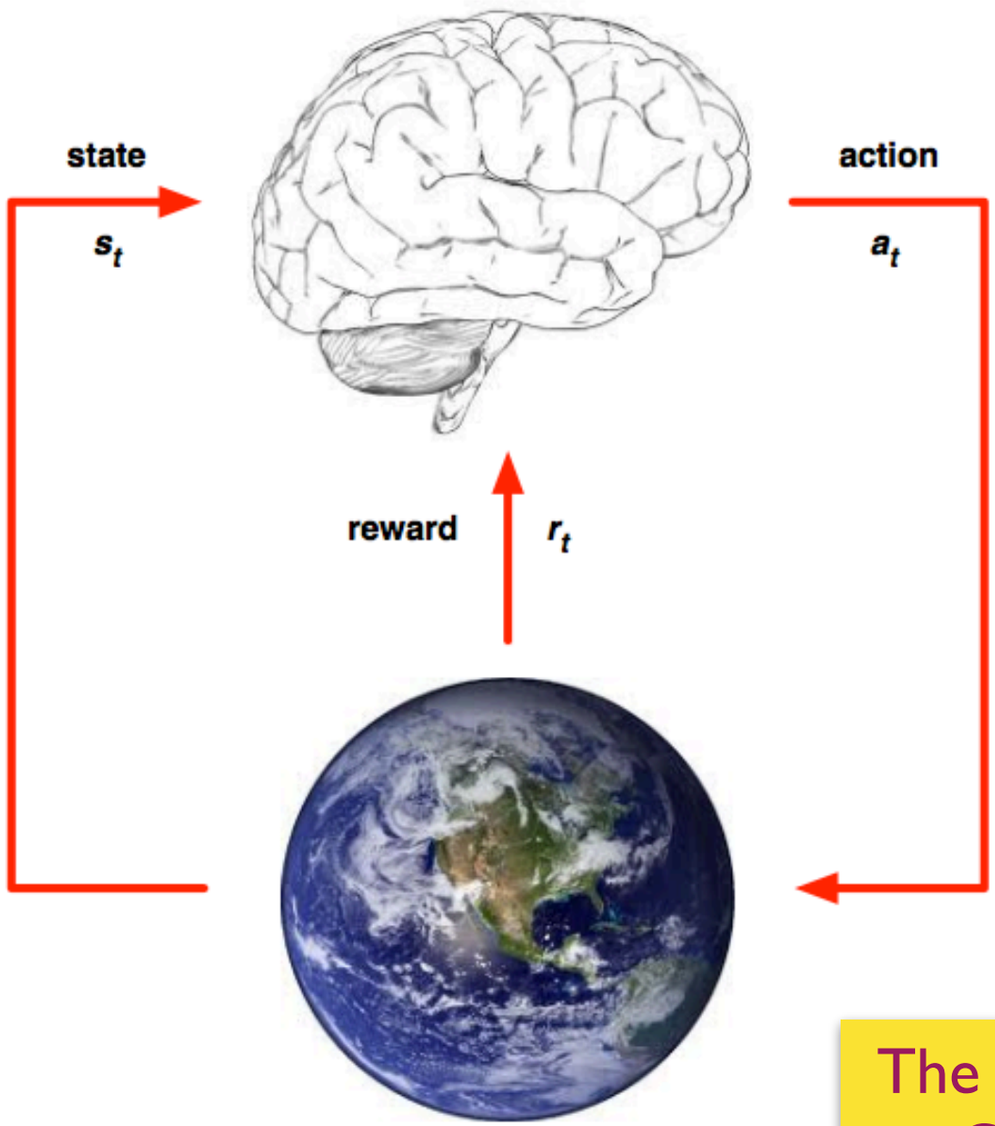


- At each step t the agent:
  - Receives state s_t
  - Receives scalar reward r_t
  - Executes action a_t
- The environment:
  - Receives action a_t
  - Emits state s_t
  - Emits scalar reward r_t

- The target is still to learn the optimal value function.

Slide courtesy: David Silver

Agent and Environment



- At each step t the agent:
  - Receives state s_t
  - Receives scalar reward r_t
  - Executes action a_t
- The environment:
  - Receives action a_t
  - Emits state s_t
  - Emits scalar reward r_t

The agent can only interact with true environment.
Can not use model for searching or planning.

- The target is still to learn the optimal value function.

Slide courtesy: David Silver

Basic Idea

- Value function based RL aims at estimating the optimal value function.

- Value function update: update using new estimation

$$Q_{i+1}(s,a) = (1-\alpha)Q_i(s,a) + \alpha\hat{Q}_i(s,a)$$

  - Monte-Carlo RL — Estimate by sampled trajectories

  - Temporal difference Learning — SARSA and Q-learning.

- Policy Improvement:

  - Based on new value function, with ϵ - greedy.

Monte-Carlo RL

- Given policy π_i, we can sample trajectories:

$$\{s_1, a_1, r_1\}, \{s_2, a_2, r_2\}, \{s_3, a_3, r_3\}...$$

- Then we can get empirical estimate:

$$\hat{Q}_i(s_1, a_1) = r_1 + \gamma r_2 + \gamma^2 r_3 + ...$$

Can we still update the policy in greedy?

- Update value function:

$$Q_{i+1}(s, a) = (1 - \alpha)Q_i(s, a) + \alpha\hat{Q}_i(s, a)$$

No!

- Follow the spirit of policy iteration, update $\pi_i \rightarrow \pi_{i+1}$

Exploration vs. Exploitation

"Behind one door is tenure - behind the other is flipping burgers at McDonald's."

- There are two doors in front of you.
- You open the left door and get reward 0
  $V(left) = 0$
- You open the right door and get reward $+1$
  $V(right) = +1$
- You open the right door and get reward $+3$
  $V(right) = +2$
- You open the right door and get reward $+2$
  $V(right) = +2$

$\vdots$

- Are you sure you've chosen the best door?

Slide courtesy by David Silver

Exploration vs. Exploitation (Cont.)

- In policy iteration, the policy improvement step is greedy:

$$\pi_i(s) = \arg\max_a Q_i(s, a)$$

- But for RL, since the environment is not fully known, greedy update may perform arbitrarily bad — need to allow some exploration.

- Common choice: use ϵ- greedy policy:

with prob. $1 - \epsilon$, execute as greedy

with prob. ϵ, execute randomly

- Theoretical guarantee: If the exploration vanishes, we can ensure convergence.

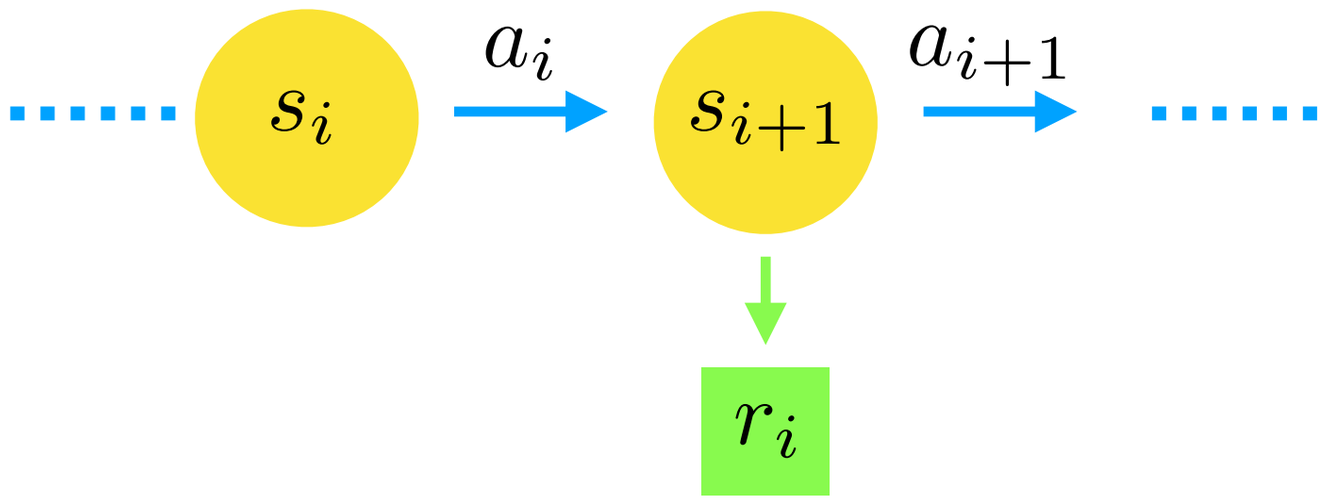
TD vs. MC

- Temporal-difference (TD) learning has several advantages over Monte-Carlo (MC)
  - Lower variance
  - Online
  - Incomplete sequences
- Natural idea: use TD instead of MC in our control loop
  - Apply TD to $Q(S, A)$
  - Use ϵ-greedy policy improvement
  - Update every time-step

Slide courtesy: David Silver

SARSA

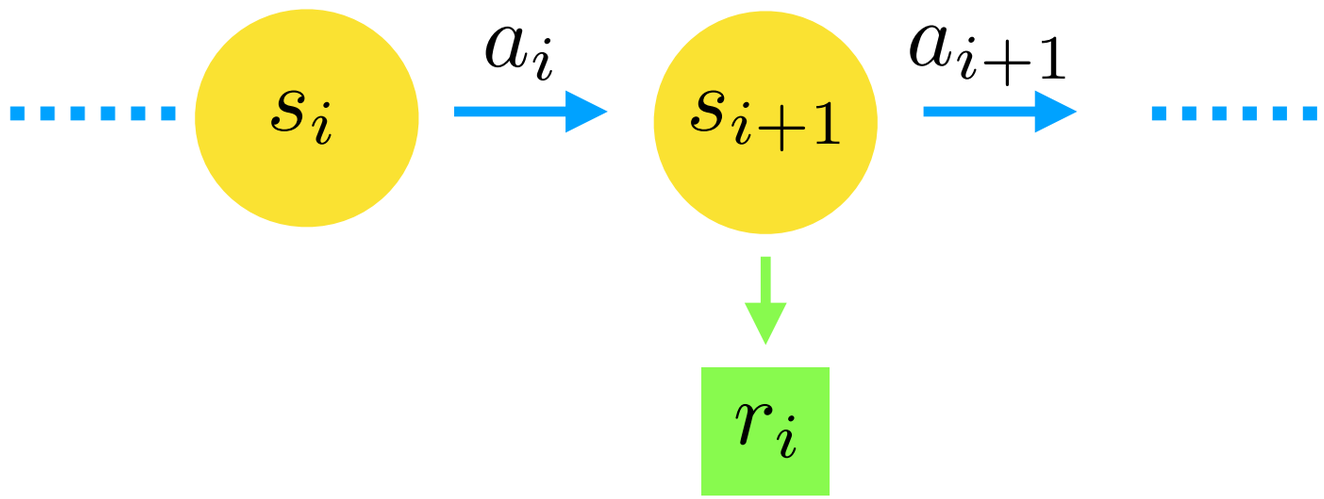
- "State-Action-Reward-State-Action" — SARSA



Run the current best ϵ-greedy policy

SARSA

- "State-Action-Reward-State-Action" — SARSA



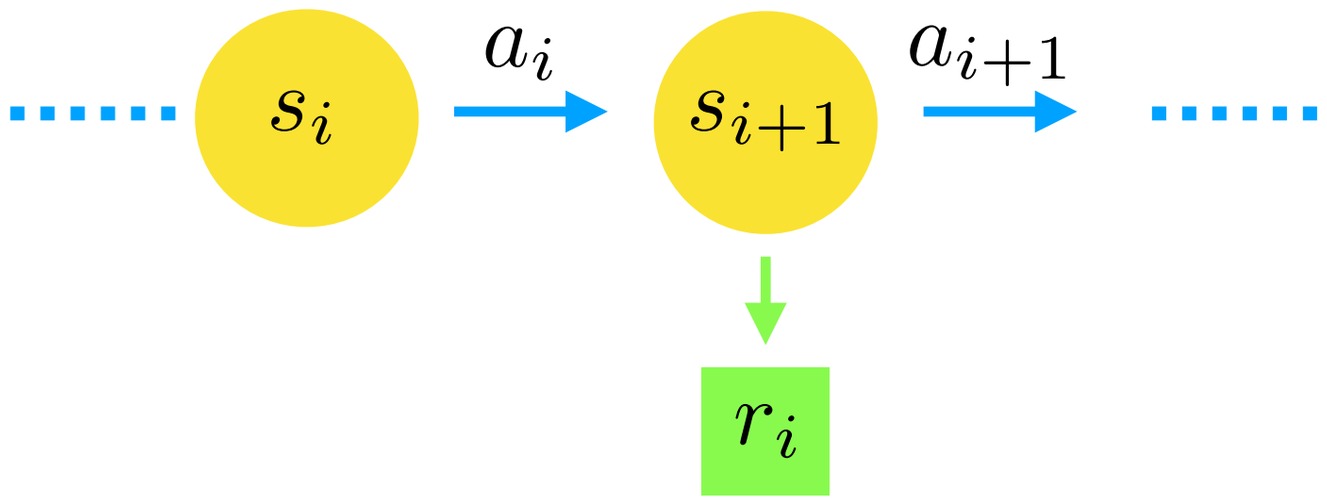
Run the current best ϵ-greedy policy

- Once collect s-a-r-s-a sample, do value function update:

$$Q(s_i, a_i) \leftarrow Q(s_i, a_i) + \alpha \Big[r_i + \gamma Q(s_{i+1}, a_{i+1}) - Q(s_i, a_i) \Big]$$

SARSA

- "State-Action-Reward-State-Action" — SARSA



Run the current best ϵ-greedy policy

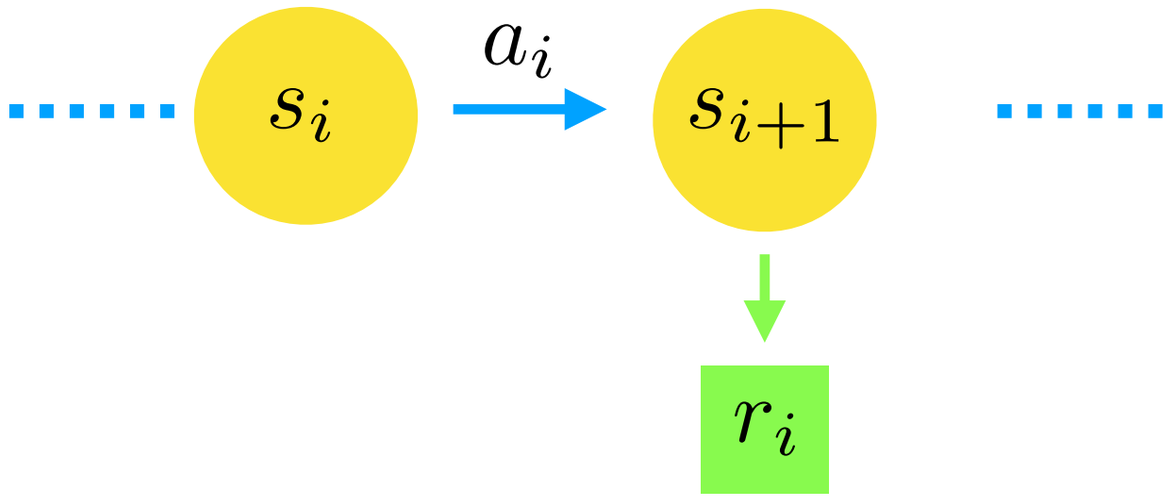
- Once collect s-a-r-s-a sample, do value function update:

$$Q(s_i, a_i) \leftarrow Q(s_i, a_i) + \alpha \Big[r_i + \gamma Q(s_{i+1}, a_{i+1}) - Q(s_i, a_i) \Big]$$

TD error

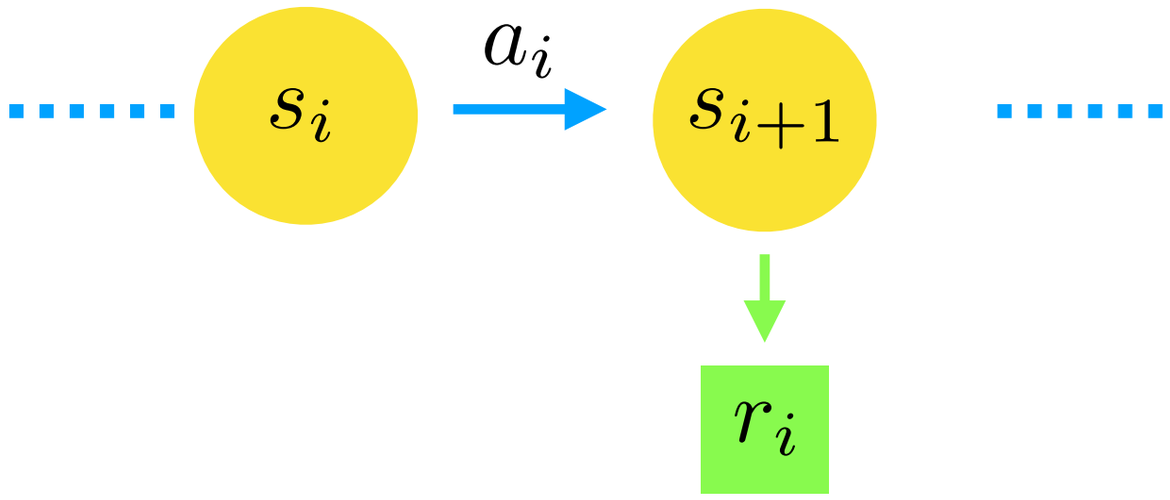
Always use the policy on-the-run, called "on-policy"

Q-Learning



Run the current
best ϵ-greedy policy

Q-Learning



Run the current
best ϵ-greedy policy

- Once collect s-a-r-s sample, do value function update:

$$Q(s_i, a_i) \leftarrow Q(s_i, a_i) + \alpha\Big[r_i + \gamma Q(s_{i+1}, \hat{\pi}^*(s_{i+1})) - Q(s_i, a_i)\Big]$$

The current best policy

The policy on the run can be different from
the current best policy in the update, called "off-policy"

Off-Policy Learning

- Evaluate target policy $\pi(a|s)$ to compute $v_\pi(s)$ or $q_\pi(s, a)$
- While following behaviour policy $\mu(a|s)$

$$\{S_1, A_1, R_2, ..., S_T\} \sim \mu$$

- Why is this important?
- Learn from observing humans or other agents
- Re-use experience generated from old policies $\pi_1, \pi_2, ..., \pi_{t-1}$
- Learn about *optimal* policy while following *exploratory* policy
- Learn about *multiple* policies while following *one* policy

Slide courtesy: David Silver

Short Recap

- When something unknown about the MDP, go RL.
- Deal with exploration vs. exploitation.
- Relationship with dynamical programming.
- SARSA and Q-learning have good theory, buy usually very slow in practice.

Policy Learning Without Value Functions

- For value function based RL, policy is not directly optimized.
  - Not capable to learn in continuous state and action space.
  - Not capable to learn stochastic policy.
  - May not learn fast.
- Can learn policy directly:
  - Parametrize policy $\pi_\theta(a|s)$ with parameter θ
    - For discrete action: softmax $\pi_\theta(s, a) \propto e^{\phi(s,a)^\top \theta}$
    - For continuous action: Gaussian $a \sim \mathcal{N}(\mu(s), \sigma^2)$

Objective Functions

- Goal: given policy $\pi_\theta(s, a)$ with parameters θ, find best θ
- But how do we measure the quality of a policy π_θ?
- In episodic environments we can use the start value

$$J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

- In continuing environments we can use the average value

$$J_{avV}(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$$

- Or the average reward per time-step

$$J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s, a) \mathcal{R}_s^a$$

- where $d^{\pi_\theta}(s)$ is stationary distribution of Markov chain for π_θ

Slide courtesy: David Silver

Policy Gradient Theorem

Policy Gradient Methods for Reinforcement Learning with Function Approximation

Richard S. Sutton, David McAllester, Satinder Singh, Yishay Mansour
AT&T Labs – Research, 180 Park Avenue, Florham Park, NJ 07932

Theorem

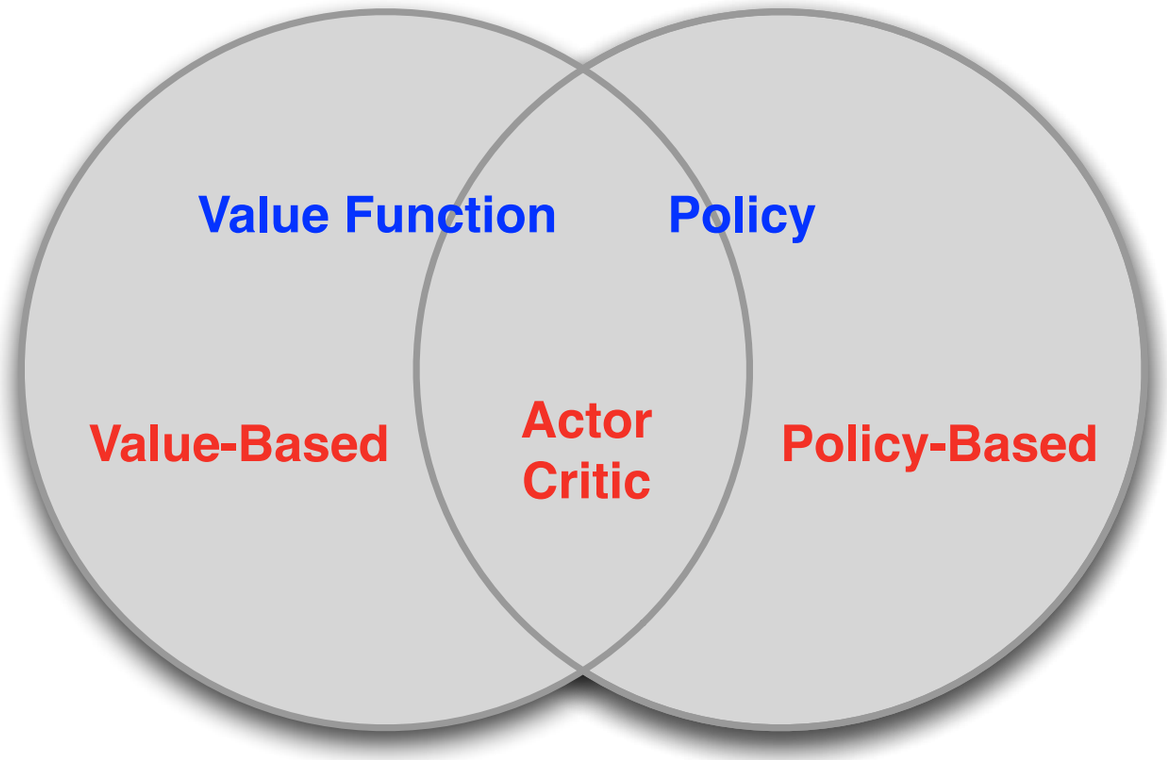
For any differentiable policy $\pi_\theta(s, a)$,
for any of the policy objective functions $J = J_1, J_{avR}$, *or* $\frac{1}{1-\gamma} J_{avV}$,
the policy gradient is

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(s, a) \, Q^{\pi_\theta}(s, a) \right]$$

Need to estimate value function

Actor-Critic

- Value Based
  - Learnt Value Function
  - Implicit policy (e.g. ϵ-greedy)
- Policy Based
  - No Value Function
  - Learnt Policy
- Actor-Critic
  - Learnt Value Function
  - Learnt Policy



Slide courtesy: David Silver

Some History (Before Deep RL)

- Early successful story of RL: human-level play of TD-gammon.
  - [Tesauro, 1995], q-learning with neural network approximation.
- But before deep RL, what RL paper usually did in experiments:
  - Grid world, mountain car, cart-pole…
- Successful stories:
  - Inverse RL [Abbeel & Ng, 2004]
  - Motor policy learning by PG [Peters and Schaal, 2008]

(b) Imitation learning.

(c) Initial reproduction.

(d) After reinforcement learning.

Decision Making: III

- Reinforcement learning (RL): background
- Markov decision process
- RL with known model
- RL with unknown model
- Take-Home Messages

Take-Home Messages

- Reinforcement learning solves decision problems by interaction with the environment.

- Markov decision process models the decision problem, when full information is known, dynamical programming can be used.

- Value function based RL utilizes MC or TD estimate of the value function.

- Policy gradient and actor-critic methods directly learn parametrized policies with SGD. Suitable for continuous state and action spaces.

Not Covered…

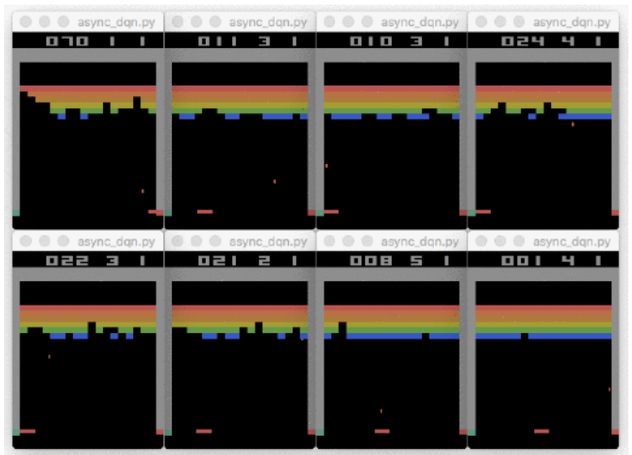
- The bandit model
- Imitation learning
- Least square policy iteration
- Actor-critic and policy gradient algorithms
- Inverse RL
- RL Theory
- …

We will come back to RL in the learning part lectures!

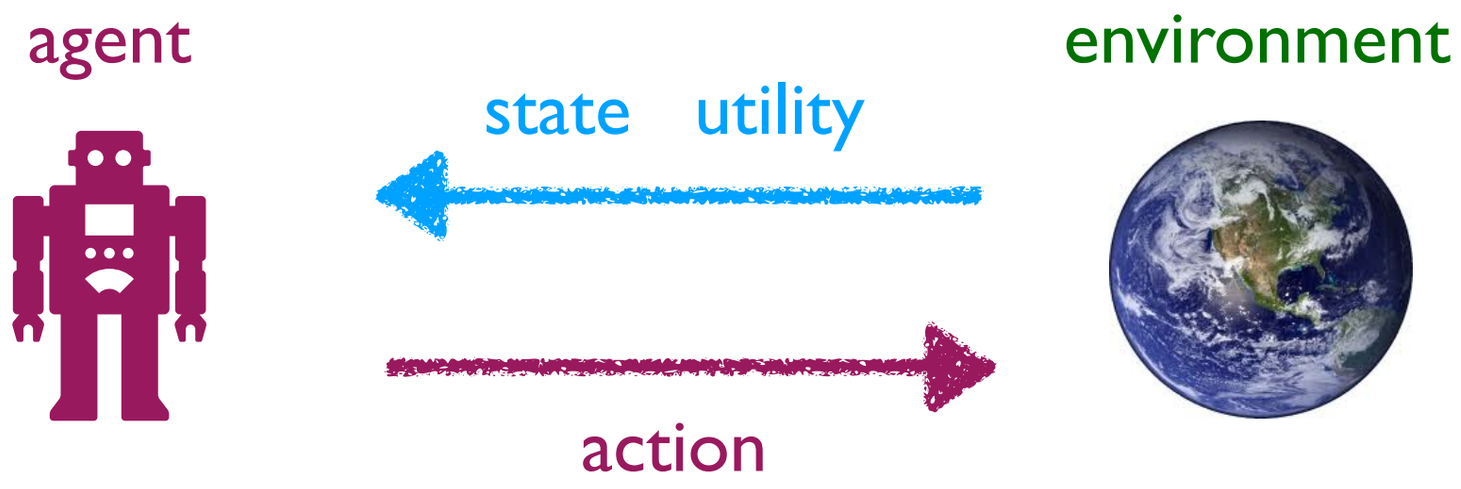
Further Reading

- David Silver's RL Course
- Deep RL course @ Berkeley
- Sutton and Barto book, 2nd edition
- OpenAI Gym platform

Decision Making

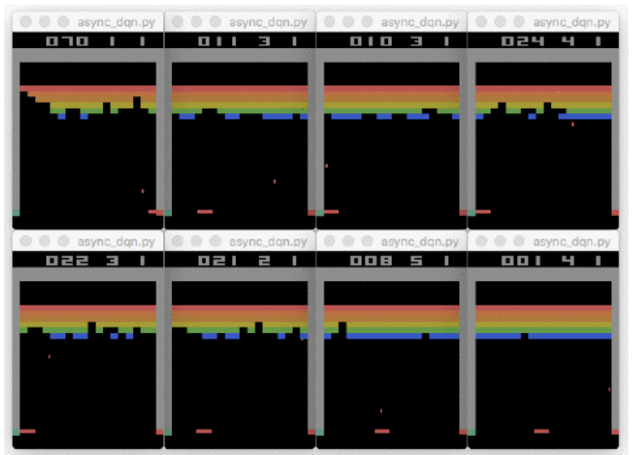


- Conduct action in any state of an environment.

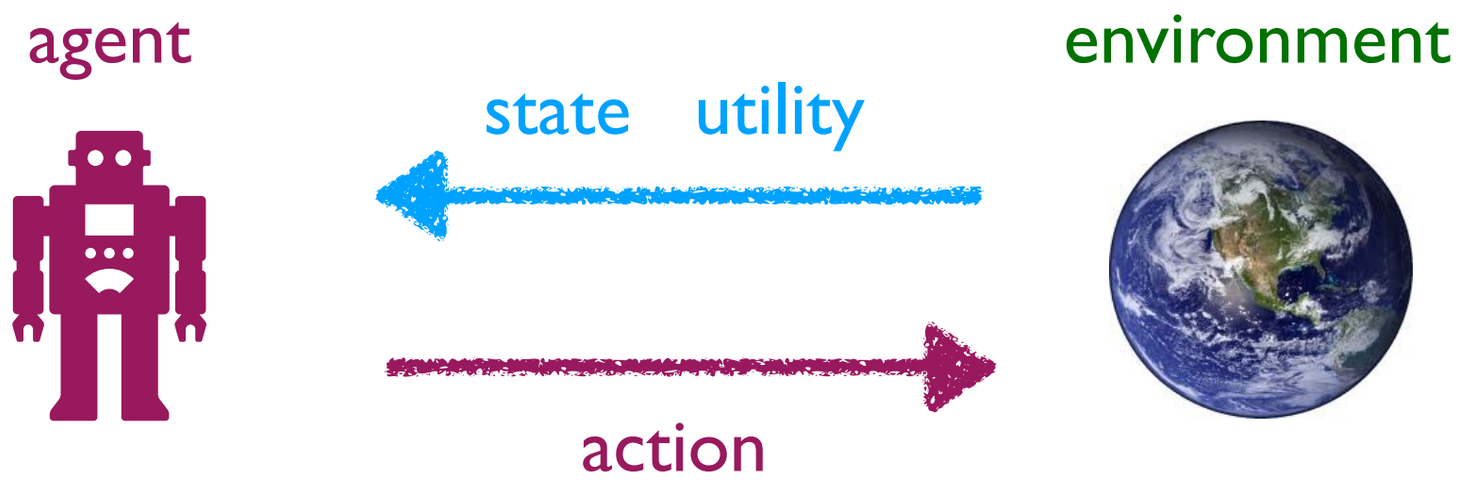


In most problems, the agent needs to do a sequence of actions w.r.t. a sequence of states.

Decision Making



- Conduct action in any state of an environment.



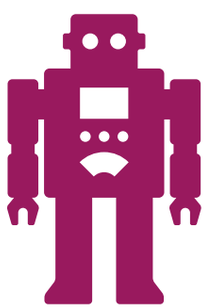
In most problems, the agent needs to do a sequence of actions w.r.t. a sequence of states.

Internal vs. External Model

We haven't discussed how to build and use the internal models!

internal model

agent



partial state

utility

action

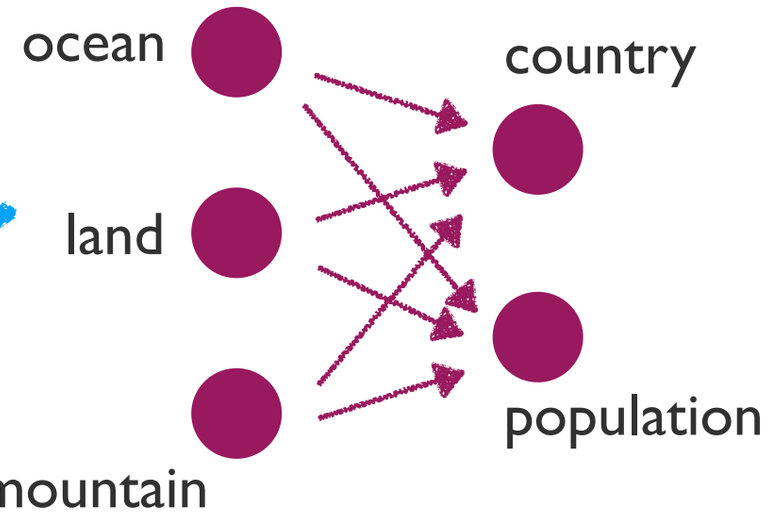
environment
external model

Knowledge Reasoning

- We will learn three kinds of knowledge reasoning strategies:
  - logic inference, probabilistic inference, causal inference.

longitude and latitude

ocean, land, mountain



See you in the next three lectures!

Thanks for your attention!
Discussions?

Acknowledgement: Many materials in this lecture are taken from
https://www.davidsilver.uk/teaching/